Authoring Tool for the Digital Classroom

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by
Sqn Ldr D Singh

to the
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
March, 1998

Certificate

Certified that the work contained in the thesis entitled "*Authoring Tool for the Digital Classroom*", by *Sqn Ldr D Singh*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. Harish Karnick)
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

March, 1998

Abstract

The growth of hypertext and hypermedia systems has led to a new way of organizing information, which is similar to the way the humans process and organize the information. In this thesis we have proposed an authoring environment which uses hypertext to develop the contents of a course which is to be taught.

In our framework a course is represented as a collection of lectures which in-turn is a collection of hyperboards. A Hyperboard is a stand-alone HTML document which can be interlinked together to form a network. This HTML document can contain text, images, and even audio.

The following tools have been implemented as the part of this thesis:-

- Tool to automate the creation of hyperboards.
- Organization tool to organize the contents of a lecture and the whole course.
- Search engine that allows the user to find the hyperboard that contains the given keyword.
- Tool to create an ON-LINE index.
- View tool that allows the user to view the contents of the course as a whole and to modify it.
- On-Line Help

Using these tools it is proposed to design a system for teaching JAVA as the first programming course for the students (This is the part of another thesis).

Acknowledgments

I would like to express my deep sense of gratitude to my thesis supervisor, Dr. H. Karnick for his expert guidance and encouragement throughout this thesis work. I am highly thankful to him to have introduced me to this exciting world of Java which was extensively used in this thesis work and also for the concept which if used may bring a revolution in the field of computer-aided teaching. I am also thankful to him for having gone through this report and structuring it in the present form.

I am thankful to the lab staff of Department of Computer Science and Engineering and Computer Center for their assistance and cooperation during this work. I would like to specifically thank Mr. Anil Kommuri,Sunil Pi ,Lt. Vivek Gupta ,Mr. Sharad Gang ,Mr. Prasad Limaye, Mr. Ambarish C Kenghe , Mr. Atul Kumar ,Mr Kshitz Krishna and all others who have directly or indirectly helped me during the course of this thesis work.

I would like to thank my wife and daughter Priyanka for their patience and support from away and near during my entire course.

Finally, I would like to thank all my friends at IIT K, who made my stay here an enjoyable and memorable experience.

Contents

List of Figures

Chapter 1

Introduction

1.1 Statement of Problem

Various kinds of teaching aids are available in the market ever since computers have entered in our daily life. These teaching aids or tutorials have been mostly designed to give a detailed knowledge of a specific subject but its understanding depends upon the student how he interacts with it and his level of knowledge etc.

It has been widely observed that there is no perfect replacement of a human instructor for teaching. Computer based teaching aids provide enough information about the subject but do not solve the problem of doubts that may crop up, during the learning process, in the mind of the student. Such a doubt can be cleared only by the human instructor. Effective teaching depends on having an effective model of a student and their learning process.

Similarly it has been observed that the instructor also certain times finds himself constraint by the amount of information present with him during the course of the class. He though would have prepared for a particular sequence to follow, but many times this sequence is either broken or he has to answer certain doubts for which the ready material may not by present in hand. It would be most suitable for him if these material is available to him as and when required on-line.

1.2 Motivation

In the light of the above discussion, it is proposed to introduce and implement the concept of a DIGITAL CLASSROOM where teaching is done by a human instructor using the computer based teaching aid. This aid is basically meant for the instructor to use during classroom instruction. It helps him to prepare a series of lectures which he can project in the class and also modify them on-line if desired.

We have proposed an authoring environment which aids in the preparation of the course material, to be used by the instructor while delivering such lectures. The authoring environment provides tools for structuring, and presenting the information to the students.

This authoring environment provides tools to organize and present textual information and also animate important concepts. For example while teaching in a mathematics class the instructor wants to shown how a particular function will behave if integrated, tools can be provided which can actually show the results of the integration in the form of animation. This helps the student to quickly understand the concept. These animation tools can however, be both general as well as specific to the subject.

In our framework a course is represented as a collection of lectures which in turn is a collection of individual hyperboards. A hyperboard is a stand-alone HTML document which can contain text, image and even audio. A single hyperboard forms a single node which cannot be further decomposed. These nodes are linked together to form a network that constitute a course.

As a first step in this direction we have designed an authoring tool that can help the author to organize only the textual information. Tools for animation can be designed as a future improvement to this thesis.

The authoring environment has been developed using the JAVA programming language. The programs written in JAVA are portable across a wide variety of platform having the java runtime environment.

1.3 Thesis Organization

Chapter 2 discusses about the concepts in authoring, various approaches used in authoring and how an authoring tool can be evaluated.
Chapter 3 discusses the various issues involved in the design of our authoring tool.
Chapter 4 discusses the implementation details.
Chapter 5 concludes the work and suggests further extensions that can be made to our present system.

Chapter 2

Authoring Tool - An Introduction

In this chapter we will discuss the concepts in authoring and the approaches used for authoring.

2.1 What is an Authoring Tool

According to [iaAt], an authoring tool is something that helps a user to develop front ends for applications, cuts development time in doing so, and increases the user's efficiency due to ease of use. A more common definition would be a software package that can create multimedia presentations, training courses etc. through a scripting language and an intuitive icon-based construction environment.

Authoring is the process of creating, structuring and managing the information that is to be presented. During the authoring process the information is first gathered, then the information is structured or organized.

2.2 Structuring the Information

The various factors that [AGR95] affect the design of information structures are :-

- Information retrieval, accessibility and navigability.
- Information security.

- Information reuse and maintainability.
- Relationships between information units and
- Layering of differing viewpoints or organizations on the information base.

The structuring technique breaks the information into atomic blocks. These blocks are called as nodes, hyperboards in our model, it contains an item of information that can not be further broken down. The next stage is to structure these atomic blocks in a more complex pattern, which involves interlinking of the blocks. Here the author identifies the anchor points within each node that forms the starting point for links to other nodes.

Information can be organized using three types of structures :*linear,hierarchical, and graph or network.*

2.2.1 Linear Structure

This structure is useful in retaining the sequential structure of the original information. To impose such structures the author can create a set of nodes that are linked together only in a sequential manner.

2.2.2 Hierarchical Structure

This structure is similar to how books are organized. Books are usually organized in the form of chapters with sections and subsections. In this type of structure each node in the system is a part of a single hierarchy and the root node is the starting point. We can start at the root node and reach any other node in the hierarchy.

2.2.3 Graph/Network Structure

Unlike the linear and hierarchical structures, this structure aims at linking nodes that linked to each other in some semantically significant sense. This helps in accessing the same information from different contexts.

The information structure chosen depends on the application in hand.Linear structure best fits training material,hierarchical structure text books, and graphical structure can model encyclopedia's.

2.3 Approaches to Authoring

2.3.1 Program based Approach

Here the application [AGR95] are developed by coding from scratch using a scripting or programming language like 'C'. This approach is used for creating very specific applications. The main disadvantage is that a lot of effort and time is spent in creating and hand-crafting each application.

2.3.2 Screen based Approach

In the screen based approach [AGR95], every screen is hand-crafted and manually linked to build the application. Emphasis is placed on the screen layout. Each screen tends to convey a single theme (a chunk of information), such as a particular topic or subtopic. Screens can be fixed or scroll-able. These screens are linked together based on the structure developed during the design phase. This approach also takes much time, as each screen must be individually designed, and thus does not suit rapid development of large systems. Pre-designed screen layouts can simplify the process.

2.3.3 Information centered Approach

In a information-centered approach [AGR95] the authors first gather the information and then structure it using standard markup languages like SGML,HTML etc. A presentation system or browser is needed to view the information. The major advantage of using the information-centered approach is that the process of structuring the information is clearly separate from the process of presenting the information. This approach does not concentrate much on layout issues. WWW (World Wide Web) is an example of a system using the information centered approach. Here the

information is encoded in HTML format and can be viewed using a presentation system such as Netscape.

Another thing that authoring tools handle differently is how they interact with the underlying operating system and graphics packages. The goal of all authoring tools is platform independence which can be obtained by using a layered or a emulated model.

2.3.4 Layered Model

A product [App],that uses a layered model will use native GUI-building toolkits to get the native look-and-feel on a platform. For example, an authoring tool that uses the layered model on a PC would build its interfaces with direct calls to Windows libraries. This is advantageous because the authoring tool does not have to provide as much software and is therefore generally less expensive. A disadvantage of this is that it is generally slower. In addition, it is impossible to get the look-and-feel of another platform other then the one it is being run on. However, it will always have the most current look-and-feel of that platform.

2.3.5 Emulated Model

An authoring tool [App],that uses the emulated model will avoid native toolkits and provide all the functionality itself. This is a more complicated way of doing things, but has several advantages. The advantages in this approach are that it does not require developers to buy extra software, it can emulate the look-and-feel of other platforms besides the current one, and it is faster because it can eliminate much of the code in the native toolkit that is not relevant to the authoring tool. However, because the vendor has to create a lot more software, the authoring tool is usually more expensive. Also, if the underlying system software changes it's look-and-feel, the authoring tool will have to come out with an update to keep current since it does not make high level calls to the current platform's API.

Which approach to use for an authoring tool chosen for any project should be based on the needs of the project. Different approaches have different advantages

and disadvantages. The layered approach is better for more economical projects that don't care as much about getting the same look-and-feel on different platforms and where speed is not as important. The emulated approach is good for projects where user customization is desired and speed is important but economy is not.

2.4 The Evaluation System

Following [Sys] are the criteria used to evaluate an authoring tool :-

- Multi-Platform Support: The tool should be able to run on several platforms, including Unix (SunOS, Irix, etc.) and IBM PC's (Windows, Windows NT). If the tool supports other platforms, such as the Macintosh, it is considered a bonus.

- Widgets / Customizable Widgets: Any tool chosen has to have a base set of widgets such as lists, scrollable windows, buttons, flashing buttons, etc. The best tool in this category would have the largest set of easy to use widgets. The toolkit must also be able to let the developers make their own widgets and integrate them into the application. An object-oriented approach with inheritance would be best.

- Interactive Builder: This is a major requirement since it is one of the main purposes of using an authoring tool. An interactive GUI builder that allows a user to "paint" the interface and then attach functionality saves development time. It can also be used for rapid prototyping and experimentation with interfaces.

- C++/dynamic-link/fork-exec/etc.: Most authoring tools have very specific functionality and to do anything beyond that, more programming must be done in another language and then utilized and interacted with by the GUI. In addition, data passing back and forth between the applications and the external programs would be good. If this cannot be done through straight C++ function calls, then it should at least be possible to accomplish this through

dynamic-link libraries or something similar. In addition, the tool should allow us to coexist with independent software libraries without interaction problems or conflicts.

- Multimedia: The authoring tool must have the capability of handling a wide array of multimedia devices and techniques, and the ability to handle a wide array of image, sound, and video formats.

- Static Graphics: Any good tool needs to let the users put in their own graphics. Static graphics, in this case, means anything that is being drawn from data. For example, a map drawn from a set of stored data points would be static graphics. In order to do this, the tool-set needs to have a good set of drawing primitives and low-level graphics calls.

- Hypertext: Hypertext can be an excellent addition to a user-interface. Without it, UI's are limited to buttons, etc. HTML format capability is preferred, but any system that would allow writing a text file with hypertext links in it and dynamically loading and using it would be acceptable.

- International Character Sets: The tool should have the capability to use international character sets. It would be preferable if the tool conformed to the UNICODE standard.

- Animation: It would also be helpful if an animation creation tool is included with the authoring tool.

- Dynamic Graphics: Dynamic Graphics are much like Static graphics (see above), but after they are created, can be manipulated and controlled via user interaction.

2.5 Related Work

In this section, we will briefly discuss some existing authoring tools for comparison.

2.5.1 Intermedia

This system [NYD88], was developed at Brown University's Institute for Research in Information and Scholarship (IRIS). It is a tool designed to support both teaching and research in a university environment. It is both an author's tool and the reader's tool.

The system, which runs on a network of Unix-based workstations, contains four integrated applications: a text editor, a graphics editor, a scanned image viewer, and a three dimensional object-viewer.

The text editor is like any other editor with the addition of style sheets for formatting. Using the style sheets, the user can define a set of styles for a particular document (such as paragraph, title, subtitle, indented quote etc.) and apply those styles to an entity.

The graphics editor can be used to create two-dimensional illustrations by selecting tools from a palette attached to each window.

The scanned image viewer can show the images digitized through a scanner. These images can be cropped, copied, and pasted as graphics images.

The three dimensional object-viewer can convert files containing three-dimensional data points into three-dimensional representations of that data with a facility to rotate, zoom in and out, hiding some portions etc.

2.5.2 Easycbt

This system [Proa], known as Easy Computer Based teaching program is an interactive teaching tool. It has tools to create Multimedia Tutorials, Tests, Quizzes and Electronic Books.

The EasyCBT Program contains :-

Easy Tutor that helps in producing Interactive MultiMedia tutorials, slide shows and other electronic publications such as manuals. By placing simple English-like Commands in standard Text files, one can create Computer Based Training programs which contain Voice, HyperText, Graphics, Pop-Up Windows, Colors, Menus, Help Systems.

Easy Book that helps the user to create Electronic Books from standard Text files. Easy Book converts these text files into a fully customizable Electronic Book with a Table of Contents Menu, Help System, HyperText, Graphics, Voice, Book-Marks, Search and Print capability, and custom VGA Fonts.

Easy TestMaker helps in creating MultiMedia multiple choice Tests. One can incorporate Graphics and Voice. Test results can be saved to disk or encrypted, allowing Easy TestMaker to be used for 'distance education' where the student takes the test and returns the encrypted Results file for scoring.

Easy QuizMaker that helps in creating MultiMedia drill and practice Quizzes.

Easy Reference that helps in creating MultiMedia Glossaries and Lists.

2.5.3 Eon - Knowledge Based Tutor Authoring Tools

Eon [Too], is an authoring tools for building intelligent tutoring systems. Eon includes tools for authoring all aspects of intelligent tutors, including the learning environment, the domain knowledge, the teaching strategies, and the student model. It is developed at the University of Massachusetts computer Science Department. The goal of Eon is to allow instructional designers to cost effectively build multimedia-rich cross-platform tutorials and learning environments with embedded intelligent instructional strategies. It's main thrust is in creating "knowledge based" tutorials, in which the instructional content is stored in a highly modular and re-usable form. Specifying how the content is to be sequence and presented to the student is done with generic, reusable "teaching strategies". Relationships between topics in the tutorial are specified using a concept networking tool. The system also maintains a "student model" of the mastery levels achieved for each topic, and the learning preferences for each student. Thus the content and style of instruction can be highly sensitive to the student's needs and learning history. The author of a tutorial can create an adaptive teaching strategy specifying how and when hints are given to a student, and when examples should be presented. Presentation of a given topic can differ for different situations, for example, a high-achieving student may receive a summary with two examples, while a struggling student may get a more directive tutorial on that topic.

The curriculum can be easily extended or modified to update information and theories, and new curriculum can be uploaded over the world wide web.

Teachers using the tutorials in their classroom can easily customize certain aspects of the tutorial to fit their needs (for example, by altering prerequisite relationships among topics or replacing pictures with ones more relevant to their classroom environment). They can easily build another tutorial while re-using the same teaching strategies.

2.5.4 Everest Authoring System

Everest [Prob], is a computer based teaching authoring tool. It is used to create a variety of interactive multimedia and computer-based training applications. It can be used to teach people how to operate various software packages, such as word processors, databases, etc. It has a powerful screen simulation capabilities, and can accurately reproduce functionality (input fields, mouse actions, pull-down menus, etc.) of the software.

These training applications can be posted on the internet for distance learning. It allows editing the content of the course on the fly, even on the Internet.

It can be used to prepare a large bank of interactive exam questions and then the software can choose a small group of questions from the bank, and present them to the user in random order. It can also randomly scramble possible answers on multiple choice questions to discourage sharing of correct answers. It has a built-in answer judging, testing and record keeping capabilities. It can create a SAT-style standardized exam so high school students can practice for their college entrance examinations.

While viewing it automatically determines the content the user needs for the next page, and silently downloads it while the user is occupied viewing the current one.

It also contains an Internet Simulator. With the simulator, one can test run his project at a variety of communication speeds (conventional modem, ISDN, DirecPC, intranet, etc.) to observe its performance.

2.6 Our Approach

In our model of authoring we have followed a mixed approach. Our model follows the Network/graph structure by virtue of which any node can get linked to any other node. This helps in accessing the same information from different contexts.

For generating the contents of the nodes we have followed both the information-centered as well as the screen based approach. The contents are obtained from the existing information and is structured using the standard markup language such as HTML. The whole information is divided into nodes which are essentially screens containing the HTML document. Thus the author during the authoring process is generating individual screens, which we call hyperboards, and then link them into a graph. The layout of the individual screen is totally flexible and can be designed by the author. Special tools are provided to automate both the layout and content generation process.

Since each node or hyperboard is essentially a HTML document so our presentation system is any internet browser e.g. Netscape, through which each screen can be viewed.

Ours is a *layered model* that uses native GUI building toolkits to get the native look-and-feel on a platform. This has reduced development time and it is also easily adaptable to any up-gradation of the lower level software on which our model is dependent.

The entire authoring tool is written in JAVA which inherently supports multiple platforms. Our tool provides a base set of easy to use widgets such as lists,scrollable window, buttons etc. Our tool also provides a GUI builder that allows a user to paint the interface and then attach functionality to it.

Since the basic content language used for individual hyperboards is HTML so it supports hypermedia which is both hypertext and multimedia. The contents of each hyperboard can contain both text, video and also sound.

Java supports UNICODE so our tool also supports international character sets for many different languages.

The tool contain a full-fledged ON-LINE help that helps the author at all stages of development.

An animation tool is presently not provided in our model but can be taken up as a future improvement. This would require a good set of drawing primitives to be incorporated in our model.

Chapter 3

Design Issues

In this chapter, we will briefly describe the high level design of the implemented authoring tool. We will discuss the functionality of our system and a justification for each function.

There are three main issues to be resolved in the implementation of the authoring tool. First, how the individual hyperboards can be created, second how these hyperboards can be organized by linking them through different types of links and third how these hyperboards can be viewed through the presentation system. The overall view of the whole authoring system is shown in figure 1.

3.1 Problem of Creating the Hyperboard

This is the main implementation issue in the design of our authoring tool. We have to provide the author or user, various techniques by which he can create hyperboards. HTML is the content language for each hyperboard. This language allows the author to incorporate text, images, and even audio into each hyperboard. We could have used any standard HTML editor for this job for our author to generate the contents of each hyperboard but that would have meant that every novice user had to first understand the inticracies of the hypertext markup language. To avoid this we decided to automate the creation of each hyperboard.

We wanted to incorporate the concept of a stylesheet that would allow the author

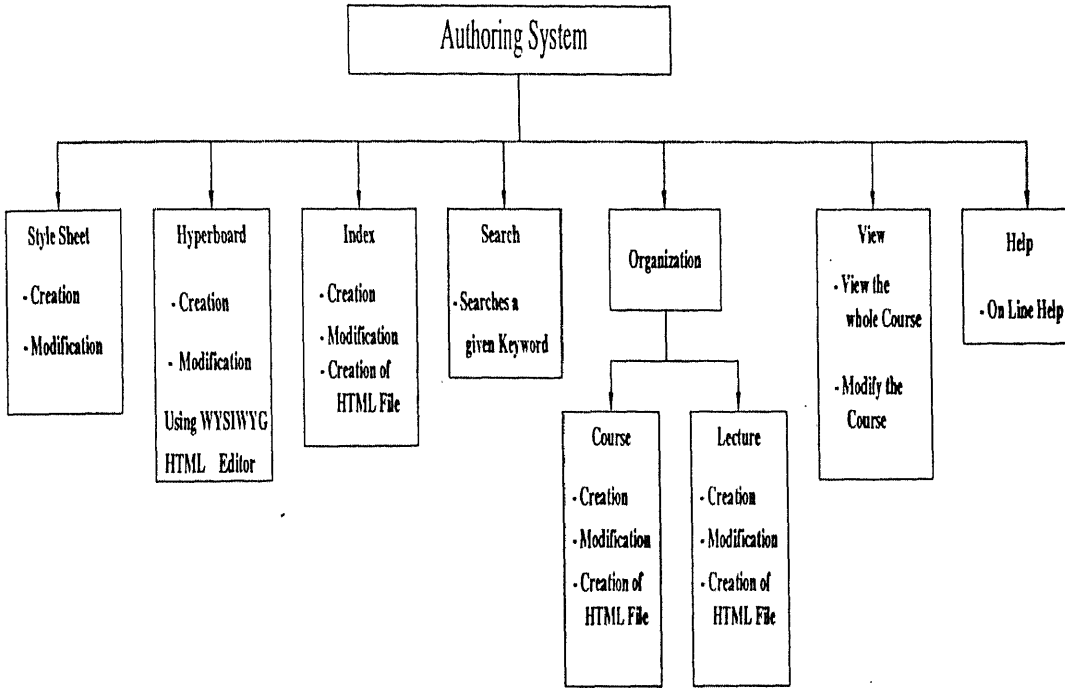

Figure 1: Main Design

to standardize the layout of various hyperboards that deals with the similar concepts and themes. A stylesheet should not only fix the layout but also generate the HTML tags for the text keyed in by the user. The generated HTML document should be such that it should not require any further modification while viewing it through any presentation system.

Creation of a stylesheet should have an easy to use interface. The creator should be given an empty blackboard where he can place the various component he desires to have in his stylesheet. These components can be simple text strings, text-fields if only a single line of text is to be keyed in, suitable for titles, or text-areas for keying in large chunks of data. The user should use the mouse interface to place and resize these components in the blackboard.

The properties of the text to be keyed in such as *font, style, color, alignment etc.* which are valid in HTML for a particular component should be decided during the placement of the component in the blackboard. These properties will be used in generating suitable HTML tags for the keyed text during the generation of the

hyperboard. Certain paragraph styles such as *normal, address, definition title, definition text etc.* should also be decided for each text-area component, which allows the user to key-in more than one line of text.

These stylesheet should be generated and saved as a file to be used later during the actual creation of the hyperboard. During the creation of the hyperboard the user should select the stylesheet and then key-in the text for the hyperboard.

Since the stylesheet can have only a limited HTML capabilities it is also required that the user should be given some facility to incorporate more complex HTML features in his hyperboard. For this it was decided that some standard stand alone HTML editor, preferably WYSIWYG, should be used. The user can design his basic hyperboard using the stylesheet and then more complex features, if required, can be incorporated using the standard HTML editor. A direct call to the editor should therefore be provided for the user in the system.

3.2 Index

An index forms a very important part of the book as it provides a non-sequential or direct access for the particular topic of interest to the reader. On similar lines it was thought that an index for a particular course will be useful for the user. The keywords in the index should be arranged in the alphabetical order with a provision for multiple references for a particular keyword. This index should also have the provision for adding a keyword or multiple words under any other keyword. This will allow the user to view or understand a particular topic or keyword under various contexts. Reference for a particular keyword, in our context, refers to the hyperboard that contains that keyword. This is analogous to the book index that contains the page number for the keyword. So the page number in our context refers to the hyperboard.

Adding a reference for a keyword involves adding the path of the hyperboard that contains the keyword, to that keyword. It was also thought that this tool should provide direct calls to the presentation system such as Netscape for the user to view the hyperboard before adding that as a reference to the keyword, in order to make

sure that the referenced hyperboard is the correct hyperboard. Similarly direct call to the HTML editor is also provided, in case the user needs to do any modification to the reference hyperboard before adding its path to the keyword.

Finally an option is provided in the tool by which an HTML document for the index can be generated directly. This is so because this index will not be very helpful if it cannot be viewed through a presentation system, as the presentation system is the main tool which will be used to view the generated hyperboard. This generated HTML document can be directly viewed through the presentation system. Thus the index also becomes a type of hyperboard.

Since the index is an ever-growing part of the book similarly our index should be modifiable after it has been created. This modification may include both addition as well as deletion of keywords.

Certain editing features such as CUT, COPY and PASTE are provided in the tool. CUT option is used to remove any particular keyword from the index while COPY and PASTE options are used to move or make a copy of any keyword from one place to more than one place.

3.3 Search

As discussed in chapter 1, it has been seen that often the instructor finds himself constrained with the amount of information available with him during the course of the lecture. Sometimes it is necessary to answer questions or clear doubts raised in the class, for which the material is not readily available in the current lecture. It would help if he can get that material on-line, as and when required, even when it is not the part of the prepared lecture.

To provide such a facility it was decided that a searching tool should be incorporated in our authoring system. This tool should help the user to locate the hyperboard that contains a particular keyword or topic. On finding the required hyperboard the user can show it directly to the student and if required can also connect it to his lecture using the organization tool described in the next section. This however, may require him to quit the presentation system and use the authoring tool to do the job.

Connecting the searched hyperboard to his current lecture strengthens his course contents making it more and more strong and will be useful when the same course is taken again. This also helps any other instructor to understand the various detours that the course took when it was last taught. So that he can prepare himself well for all kinds of standard doubts on a particular topic.

This concept makes the whole system quite open, as now hyperboards of diverse subjects and fields can be interconnected. To cite an example, assume that while teaching the subject of theoretical computer science the instructor needs to explain a concept of mathematics to the student, with this tool he can search for the hyperboard that contains that topic in the mathematics course and then can either connect it to his current lecture or just show it to the students and proceed further, all this can be done by a few keystrokes.

The user interface of the search tool has a text-field that allows the user to key-in the text or a keyword to be searched. He has the option to search this piece of text either in a single hyperboard if he knows the hyperboard name or a lecture file that contains a set of hyperboards or a course file as a whole which contains a set of lecture files. Options are provided for the user to either key-in the hyperboard or lecture file or course file name or search for the names using the file dialog box. Option are provided to perform the search in both lower and uppercases.

The result of the search yields the name of the hyperboard or the list of hyperboards that contains the particular piece of text, to be searched, along with the line number and the column number that contains the text. A direct call to the presentation system such as Netscape is provided for the user to view each such hyperboard. Similarly a direct call to the HTML editor is also provided, in case the user needs to do any modification on the searched hyperboard.

The searched hyperboard can be shown to the students during the class and can also be connected to the lecture using the organization tool described in the next section.

3.4 Organization

This tool is also an important part of our authoring system. This tool helps the user to organize his lecture as well as the course. As already explained in the previous chapter, lecture in our context is a file that contains a list of hyperboards, and a course is a file that contains a list of lecture files. After the individual stand-alone hyperboards have been designed or created using the stylesheet, they should be organized in such a way that a group of them forms a lecture and then a group of lectures forms a course. This organization tool separates the process of hyperboard creation with that of linking. The user at the time of hyperboard creation need not bother about how when and where his hyperboard would be linked. This makes the whole process highly flexible as now a hyperboard can be linked at more than one place in totally different courses. There can be circumstances in which one or more hyperboards are not separable even in such cases this organization tool should be used to form a sequence. The sequence of individual lectures in terms of hyperboards is decided at this stage.

3.4.1 Organization of a Lecture / Course

Organization of a lecture involves creation of a file in which names of the hyperboards are added. Adding the name of the hyperboard, in our context, refers to the path of the hyperboard in the system. Option are provided for the user to locate the hyperboard in the system using the file dialog box and then add its path to the lecture file. Once the user locates the hyperboard file, addition of its path to the lecture file is automatic.

Editing features such as CUT, COPY and PASTE are also provided in the tool. CUT option is used to remove a particular hyperboard from the lecture file while COPY and PASTE option is used to move or copy the name of the hyperboard from one place to more than one place within the lecture file.

Finally an option is provided in the tool by which an HTML document for the lecture file can be generated directly. This generated HTML file can be directly viewed using the presentation system. Thus the lecture also becomes a hyperboard

and has all the properties of our general hyperboard. The lecture is converted into an HTML document because that is the only type of document available for viewing.

Organization of a course involves creation of a file in which names of the lecture files are added. Lecture files are the HTML document generated during the lecture file organization. Adding the name of the lecture file, in our context, refers to the path of the lecture file in the system. An option is provided for the user to locate the lecture file in the system using the file dialog box and then add its path to the course file.

It also contains the features such as CUT, COPY and PASTE similar to the one present in the lecture organization tool. Finally it also has the option to generate an HTML document for the course file, as present in the lecture organization tool, which can be directly viewed using the presentation system.

3.5 View

Organization of lectures and the courses as explained in the last section though an important function would lack completeness, unless a tool is provided to the user through which he can view the whole course at one stroke and also modify it if required. This would give a complete picture of the whole course to the user, where it is possible to see the list of lectures in each course and the list of hyperboards in each lecture. Along with this the user is given an option to reorganize the lecture or course by moving the hyperboards from one location to another.

It was decided that the interface for this tool should look similar to the one that Windows-95 uses i.e. Windows Explorer. The user is given an option to pick up the course he wishes to view. For this a file dialog box is appropriate for the user to navigate through the system's directories and pickup the course of his choice. The whole interface window is divided into two frames, one showing the course contents .i.e. the lectures and other frame showing the contents of the hyperboard, the user wishes to see.

After the user selects the course to be viewed, the contents of the course i.e. the lecture files in it is shown to the user in the form of a tree with a "+" sign

appearing before the name of the each lecture file indicating that it contains a list of hyperboards. A mouse interface is provided in this frame in such a way that when the user clicks the mouse button on a particular lecture filename, it explodes into a list of hyperboards that are present in it. The "+" sign now changes into a "-" sign, on clicking the mouse button on the "-" sign the original state of the lecture file appears.

The contents of the hyperboard can be viewed in the second frame. For this facility a direct call to the presentation system such as Netscape is provided. The default window of Netscape is not within the user's control so a *java-script* is written to open the Netscape window within the second frame. Similarly a direct call to the HTML editor is also provided for the user to modify the contents of the hyperboard in case he wishes to do so.

There is also a need to show the contents of the whole course, because it is only during the course of viewing the contents of the course that a course is likely to be modified. This may involve either removing a full lecture or any hyperboard within the lecture or even moving or copying a hyperboard from one lecture to another. Thus, we require options like CUT, COPY and PASTE. These options will allow the user to edit the contents of the course. CUT option allows the user to remove any hyperboard from the lecture file while COPY and PASTE option allows the user to copy or move a hyperboard from one lecture file to another.

3.6 Help

On-line help is an important feature of any software package. An on-line help tool is a part of our authoring system. This on-line help provides a help manual on each tool separately giving a step-by-step procedure on how to use the tool and any special care that must be taken while using the tool.

Apart from this, definition of the various keywords that are used in the system is also provided in the system. An on-line query system is provided by which the user can query on various aspects of the system such as keyword location, usage of the system, various techniques that can be employed etc.

3.7 Scripting/Visual Languages vs. Code Libraries

There can be two different approaches [lvCL] that can be employed for building the Graphical User Interface (GUI) :

- code libraries
- visual/scripting languages

Each has its own advantages and disadvantages.

A code library is a library of subroutines that can be called and used from a language such as C or C++. In this thesis, java will be used as the example language. With a code library, most programming is done in java. To utilize the GUI/multimedia functionality from the authoring software, calls are made to the subroutines in the code library.

A scripting language is more ambitious than a code library in that it tries to cover all the possible functions one would use when creating a GUI. It is essentially like learning a new programming language with a lot of very specific functionality.

A code library is generally harder to use than a scripting language because the developer must know java already and then learn to use the code library functions. Significant amounts of time can be spent debugging code that has nothing to do with a GUI. With a scripting language, development is generally a less daunting task since there is a more limited amount to learn and much of the debugging has been taken care of by the scripting language itself.

However, a code library is usually much more flexible and versatile than a scripting language. A program that uses a code library has the advantage of being able to utilize all the functionality of the underlying programming language in addition to its own. Working with applications that weren't designed with the code library is easy because one can utilize the programming language itself to do it. Data passing is likewise very easy.

A scripting language can only do what it was made to do. In other words, it is limited in its functionality compared to a code library. Working with applications not made with the scripting language can be difficult and data passing is usually non-trivial.

Smaller, stand-alone projects usually work best with a scripting language because it allows faster prototyping and development. However, for projects that want to work with other applications and do a lot of things that no authoring tool can be expected to do, a code library is usually best.

In our thesis code libraries of the java programmming language is used for the development of the software.

Chapter 4

Implementation

In the previous chapter we described the high level design of our authoring system. In this chapter we will discuss the implementation details of the various tools that are present in our system. These tools have been implemented using the programming language Java.

4.1 Reason for Using Java

With the wide spread growth of the Web and the Internet the need for developing distributed software is increasing. Now we need to develop applications that can run on a wide variety of networks and computer systems without the prior knowledge of the target hardware and software platforms. This feature, architectural neutrality, is the main reason for using Java. Java supports features like portability, and architectural neutrality.

When a program written in Java is complied the compiler produces machine code for an intermediate virtual machine. This code is called the byte code. This byte code can be executed on different platforms using the java run-time environment for the specific platform.

Programs written in Java can be classified in two types: applications and applets. An application is a stand-alone program that runs from a shell or DOS prompt. An applet is a program which can be run using a java-compatible browser like Netscape

3.0, Hot Java,Mosaic etc. An applet has much stronger security restrictions than an application. Java uses a pure object-oriented approach for programming. Everything in Java is a class and nothing else. A package in Java is used to refer to a collection of classes. Some of the packages found in the java system are :

- **java.lang package** This package contains classes for creating basic data types such as integer,float,boolean,char, and stings etc.
- **java.io package** This package contain classes for managing streams like standard input, standard output,and standard error. User defined data streams, and pipes can also be created.
- **java.util package** This package is a collection of utility classes such as Directory, Hash table, Time and Date , and the Vector class.
- **java.net package** This package contains a collection of classes for accessing, and sending information over the network using sockets, and datagrams.
- **java.awt package** This package is known as the Abstract Window Toolkit. It contains a rich set of classes for handling graphical user interface components such as Buttons, Lists, TextAreas, Textfields, scrollbars, Menus etc.
- **java.applet package** This package provides classes that can support applet creation.

We have coded the whole authoring system in Java using the Microsoft Developer Studio support environment. The mainmenu is shown in the figure 2.

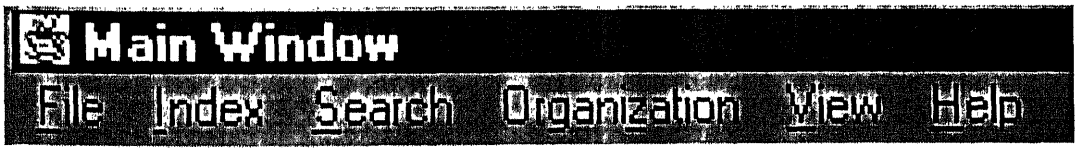

Figure 2: Main Menu

4.2 Creation of StyleSheet

In this tool all the user interfaces were developed using the Resource wizard and then were joined together in one module. The user here is provided with two main panels, shown in figure 3:

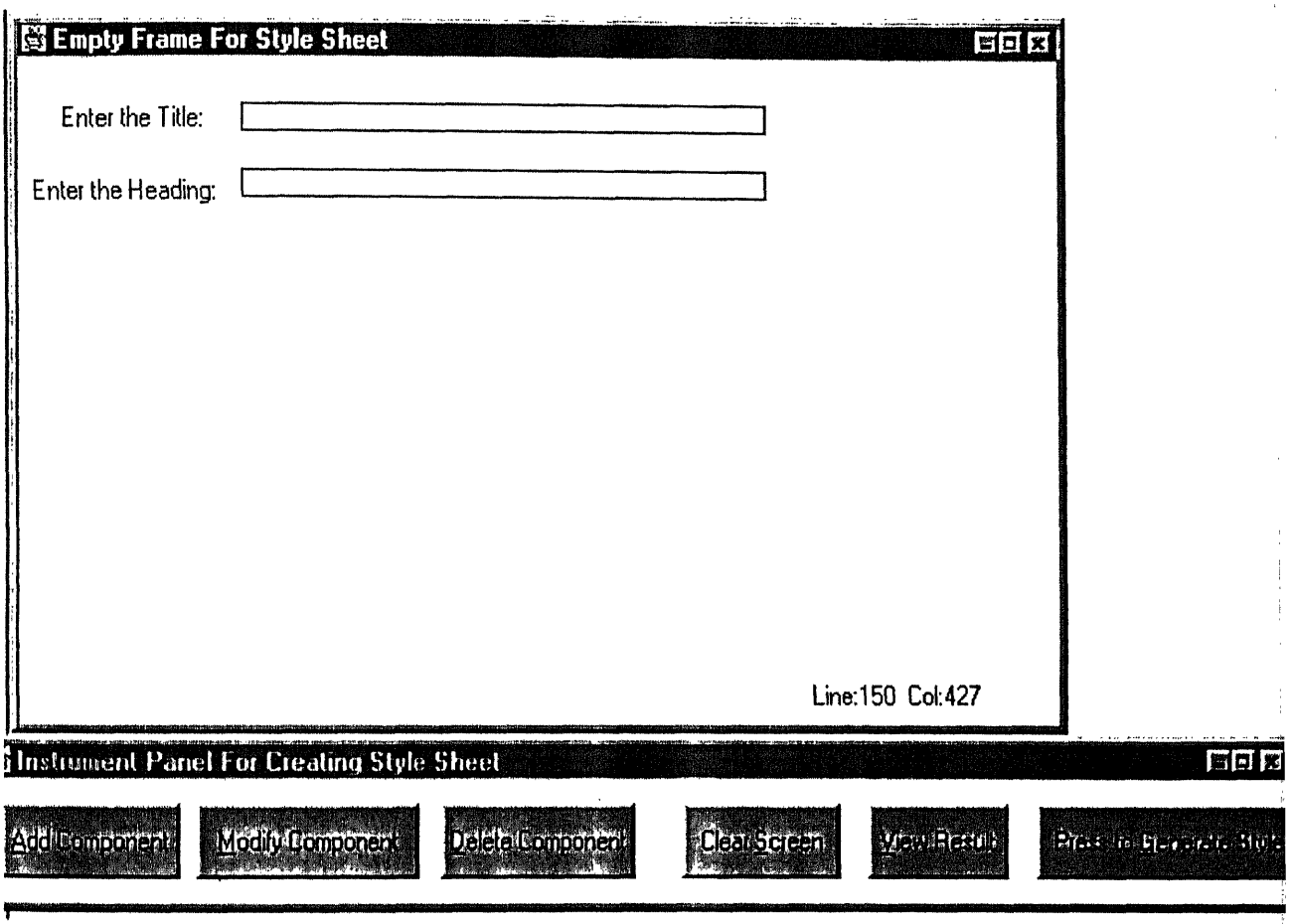

Figure 3: Creation of Stylesheet

- A Blackboard : It is re-sizable window that allows the user to place the various components such as text strings,TextArea and Textfield etc.

- Instrument Panel: This panel provides a host of buttons to carry out tasks like adding a component to the blackboard, modifying its properties, deleting the component, viewing the overall result and generating the stylesheet.

In this module the constructor creates a linked list of the following structure. All

the components mentioned above such as textstring, TextArea, Textfield etc. are the nodes of this linked list. Along with the type each node also holds the various properties of each component. Each component when created gets added in this list and when the user wishes to generate the stylesheet this linked list is written to the disk file. The main aim of this style sheet is to generate the hyperboard, which is a HTML document, hence the blackboard comes with two mandatory fields a title field and the heading field. This is done because every HTML document must have a title and the heading. The node of this linked list has the following structure :

```
component_object {
        String comp_name;
        String comp_type;
        String comp_x;
        String comp_y;
        String comp_width;
        String comp_height;
        String comp_light;
        String comp_used;
        String comp_align;
        String comp_font;
        String comp_style;
        String comp_message;
        String comp_head;
        String comp_color;
        String comp_para;
}
```

4.2.1 BlackBoard Module

This module creates an empty window for the user to place his components. This blackboard is re-sizeable so that the user can place as many component as he desires

and is not constrained by the size of the window. The blackboard shows the current position of the mouse pointer in the lower right corner as soon as the mouse enters the blackboard area . This helps the user to precisely position his components. This module uses the following sub-modules:-

- *MouseMove*

This submodule continuously monitors the current position of the mouse pointer and updates the line and column position on the lower right corner of the blackboard, see figure 3.

- *MouseUp*

This submodule stores the current position of the mouse pointer whenever the mouse button is released. If the current mode of operation is *add a component* it calls the paint module to draw the component in the blackboard. If the current mode is *move a component* it calls the paint module to erase the old position of the component and re-draw it at a new place.

- *MouseDown*

This module stores the current position of the mouse pointer when the mouse button is pressed.

- *MouseDrag*

This module continuously monitors the current position of the mouse pointer when it is moved in the blackboard with its button pressed. In case the user is drawing a component this module in turn calls the paint submodule which continuously erases the old figure and draws the new one as the mouse is moved.

4.2.2 Instrument Panel module

This module is used to create, modify or delete nodes from the above mentioned linked list.It contains the following submodules :

- Add a Component
- Delete a Component
- Modify Component
- View Result
- Clear Screen
- Generate the StyleSheet

Add a Component

On activating this submodule it opens a window showing the names of the various components that the user can add to the blackboard. The components include text strings, Textfield, TextArea and the header. A mouse interface is provided for the user to actually draw these components at the desired position and of desired size on the blackboard. The properties for these components are also decided at this stage. The properties include *style, font, color, paragraph type, alignment.* On deciding the various options the module calls the InsertElement module that forms the node for the component and adds it to the linked list. The user interface is shown in figure 4.

Delete a Component

On activating this submodule a window is opened in which the user is shown the list of the names of the currently created component. The user can use the mouse interface to select the component he needs to delete and press the delete button which in-turn calls the DeleteElement module to actually remove the component from the linked list. The user interface is shown in figure 5.

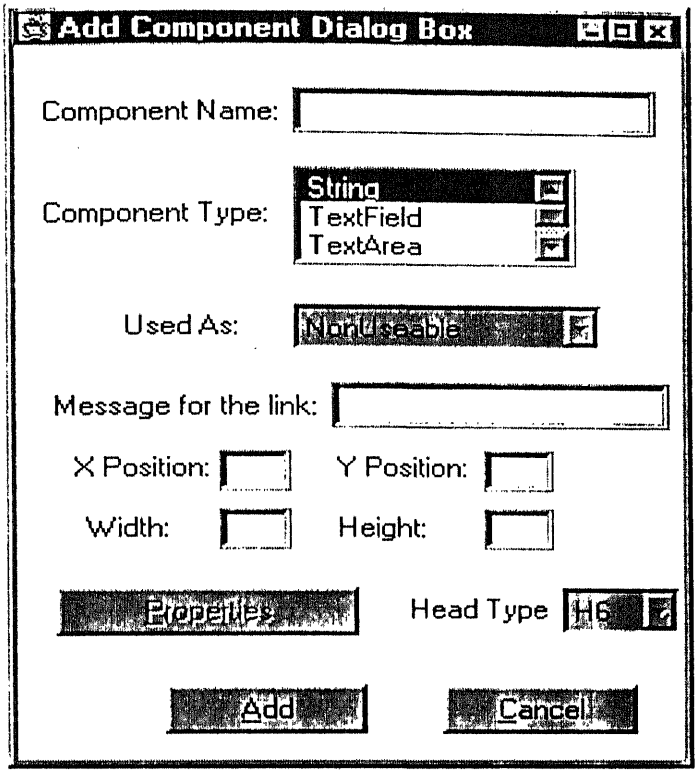

Figure 4: Addition of a Component in the Stylesheet

- *Modify a Component*

On activating this submodule a window is opened in which the user is shown the list of names of the currently existing components. On selecting the particular component, its properties are shown in the same window and the user can modify them. After the necessary modifications are carried out the user can press the modify button which in-turn calls the ModifyElement module to carry out the modification. The ModifyElement first creates a temporary node in which the old properties of the component,to be modified, are copied, necessary changes are carried out then it removes the old node from the linked list and adds the temporary node to the linked list.

This submodule also provides facility to move the component from one place to another. Whenever the user selects a component to be moved, the component is highlighted to identify and distinguish it from other components. To move a

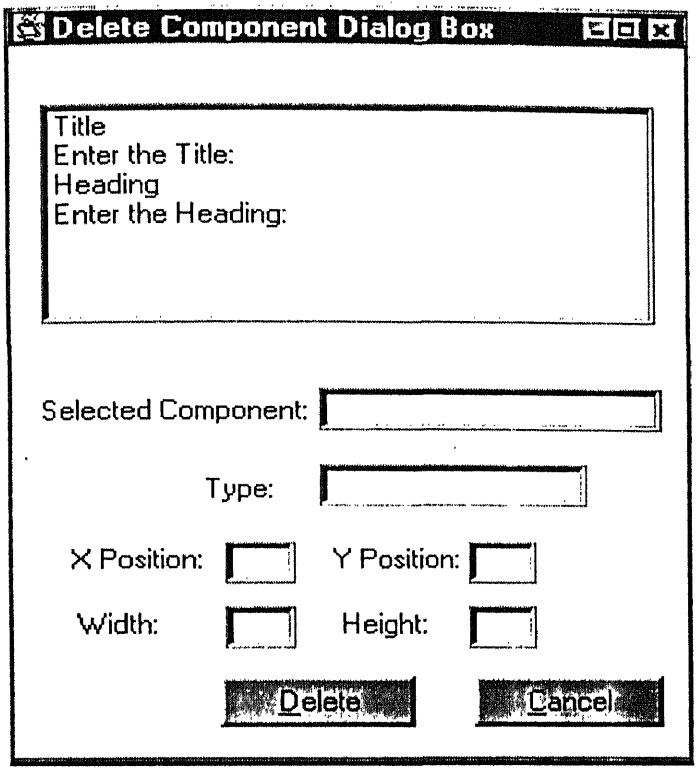

Figure 5: Deletion of a Component from the Stylesheet

component this submodule erases the old figure and re-draws it at the new place. The user interface is shown in figure 6.

View Result

This submodule helps the user to view the stylesheet the way it will appear when he uses it to create a hyperboard. This module uses its own layout to place the actual component in the blackboard. String components are placed as labels and other components as textfields and textareas which Java provides through its awt (Abstract Window Toolkit). Size and position of these components are obtained from the node saved during the creation of the component. Necessary scrolling of the blackboard is provided in case the number of components are more than what can be easily fitted in the blackboard.

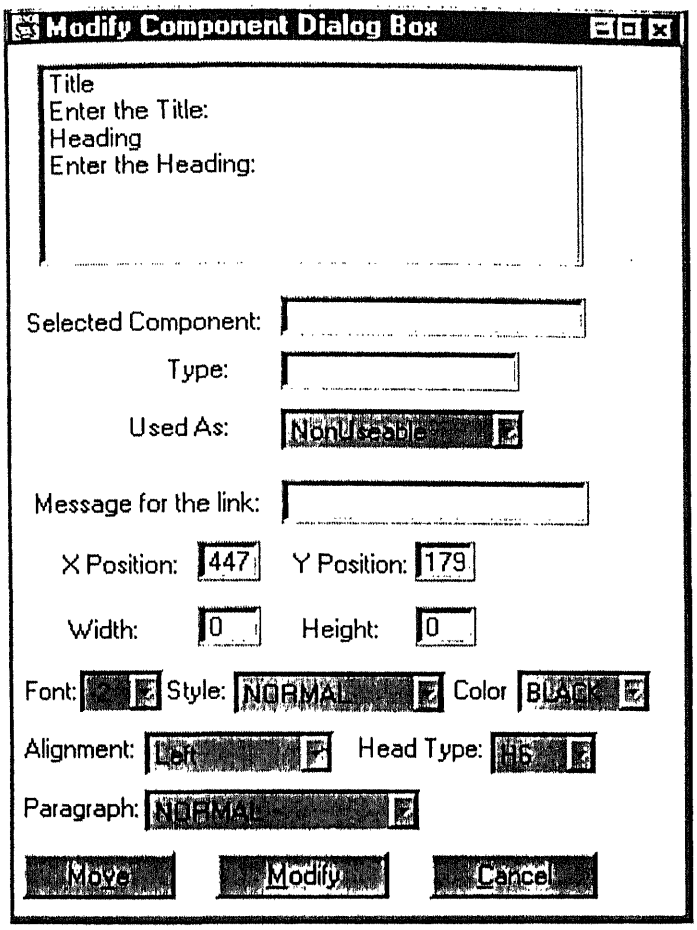

Figure 6: Modification of a Component's property in the Stylesheet

- *Clear Screen*

This submodule clears the blackboard and provides an empty blackboard. It essentially deletes the filled linked list and creates an empty one.

- *Generate StyleSheet*

This submodule copies the linked list created by the above process in a disk file. Here the linked list is traversed from start to end and each component is written as a separate line with its properties separated with a comma. Before the linked list is copied to the disk file the user is shown a file dialog box to decide the name and the directory where he wants his stylesheet to be placed. This submodule then calls the

SaveFile submodule which opens the disk file and copies the linked list.

4.3 Modify a StyleSheet

This submodule is used to modify the contents of an exiting stylesheet. It provides a file dialog box for the user to pick-up the desired stylesheet from the system's directory and then calls the overloaded constructor of the create stylesheet module explained in the previous section. This constructor creates an empty linked list for the component, opens the disk file containing the stylesheet and calls InsertIntoVector module to read the file and fill the linked list. Rest of the program is same as explained in the previous section.

4.4 Creating a New HyperBoard

This module uses the stylesheet created in the above section to provide a kind of input form for the user to create a hyperboard. Before creating the hyperboard the user has to select the stylesheet using a file dialog box. On selecting a suitable stylesheet the module opens a window displaying the components present in the stylesheet. Now the user can key-in the contents of the hyperboard and on pressing the *Generate a HTML document* button the hyperboard is created. The various properties of the components decided during the stylesheet creation are used to generate suitable HTML tags for the matter keyed-in.

This module reads the stylesheet and creates the linked list of the components present in it. Each node of this linked list has the similar structure as explained in the create stylesheet section. After creating the linked list a separate module executes and places these components in the input window. The various submodules used in this module are :-

- InsertIntoVector
- GetInfo
- Convert

- SaveFile

These submodules are explained in appendix A.

4.5 Open an Existing Hyperboard

As already explained in the design phase, our stylesheet can help the user to generate a general purpose HTML document with most basic HTML characteristics. It may happen that the user needs to add some more features to his hyperboard, that is, certain advanced features of HTML. For this we have provided a direct access to a stand-alone HTML editor for the user. With this editor the user can add all the advanced feature which he wishes to add to his hyperboard. This module provides a file dialog box for the user to select the hyperboard which he wants to edit and calls a RunCommand submodule. This submodule creates a separate process and makes a direct call to the HTML editor with the selected hyperboard as the argument.

4.6 Creating an Index

This module is used to create a book like index for the full course. This module opens two frames in the main window, one frame is used to add the keywords and the other frame is used to add the references for these keywords. A reference here means the path of the hyperboard that contains the keyword. This module also provides a facility to add a keyword under various keywords, this allows the user to understand a particular keyword in various contexts. The user interface is shown in figure 7.

The module creates a linked list of the keywords whose structure is given below :-

```
index_object {
        int number;
        String objname;
        String state;
        Vector subcomp;
}
```

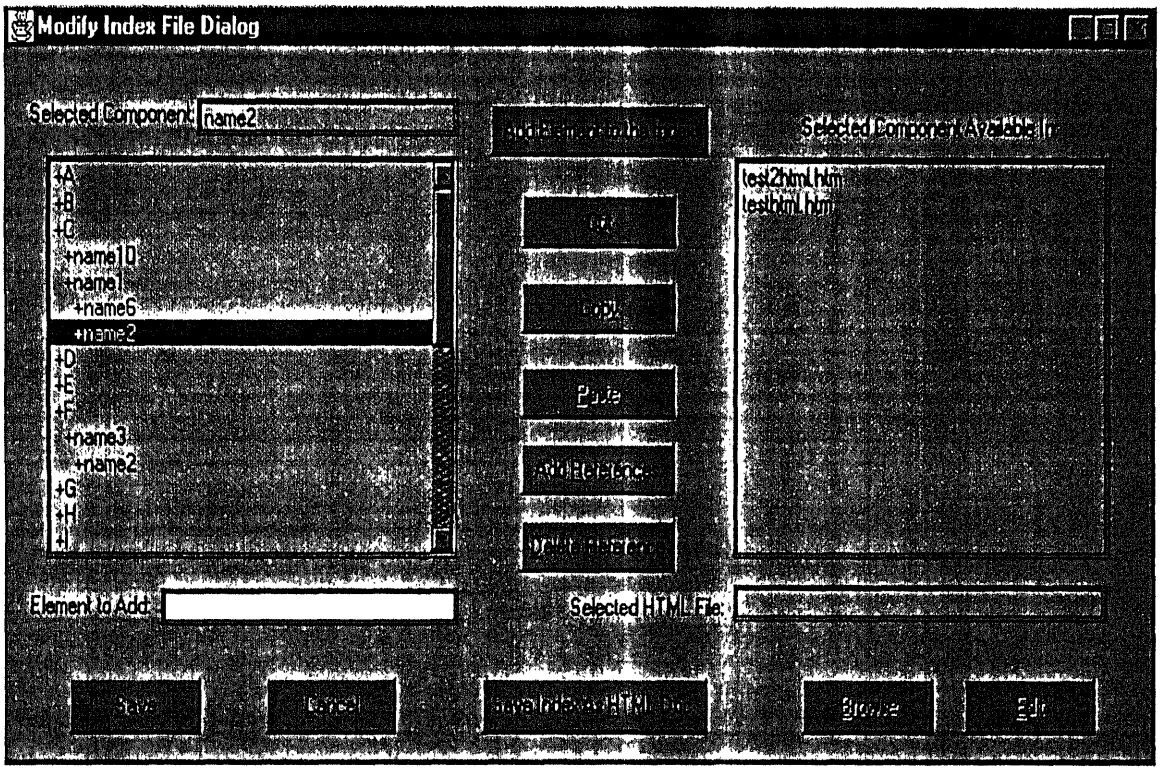

Figure 7: Creation of Index

The structure contains the keyword name as the objname, number to indicate which is the level of this keyword whether it is stand-alone or is present under some other keyword and a linked list of the references for this keyword . The structure of this linked list is given below :-

```
index_subcomp_object {
        String subdir;
        String subfile;
}
```

This linked list contains the references for this keyword. As already explained reference here means the path of the hyperboard that contains the keyword, hence

this structure contains the directory and the name of the hyperboard file. Thus the index is a linked list of linked lists.

The new keyword is added as a node to the main linked list and its references are added to the linked list connected to this node. The various other submodules in this module are :-

- Fill
- InsertMainElement
- DisplayMainlist
- InsertSubcomp
- DisplaySubcomp
- CutElement
- PasteElement
- SaveFile
- SaveHtmlFile

These submodules are explained in appendix A.

4.7 Modify Index

This module is used to modify the contents of the exiting index. It provides a file dialog box for the user to pick-up the desired index from the system's directory and then calls the overloaded constructor of the create index module explained in the previous section. This constructor creates an empty linked list for the keywords, opens the disk file containing the index and calls *fillformodify* submodule to read the file and fill the linked list. Rest of the program is same as explained in the previous section.

4.8 Open Index

This module is used to show the various keywords along with their references to the user who wants to use the index. It provide a file dialog box for the user to pick-up the desired index file and opens the window with two frames. The left frame shows the list of keywords present in the index. On clicking the mouse button on a particular keyword it's references are shown in the right hand frame.

This module uses the linked list of linked list with similar structure as explained in the create index section. It creates an empty linked list and calls the fillForView module to read the disk file and fill the linked list. Other submodules in this modules are :-

- InsertMainElement
- InsertSubcomp
- DisplayMainlist
- DisplaySubcomp

All these modulcs work in the same way as they work in the create index module.

4.9 Search

This module is used to search the hyperboard for some specific text or a keyword. The search can be carried out on a single hyperboard or on a whole lecture or on the whole course. Single hyperboard search is the fastest. The user is given checkboxes to decide where the search is to be performed i.e on the hyperboard file or a lecture or on the course file. Necessary textfield area is given to the user to key-in the name of the hyperboard filename or the lecture filename or the course filename. In case the user doesn't remember the name of the files he can press the BROWSE buttons present in front of these textfield which opens a file dialog box for the user to search the required filename. The string to be searched is keyed in a specified textfield. The search is case sensitive. Results of the search are presented in the lower half of

the window as the name of the filename that contains the specified string along with the line number and the column number. The user interface is shown in figure 8.

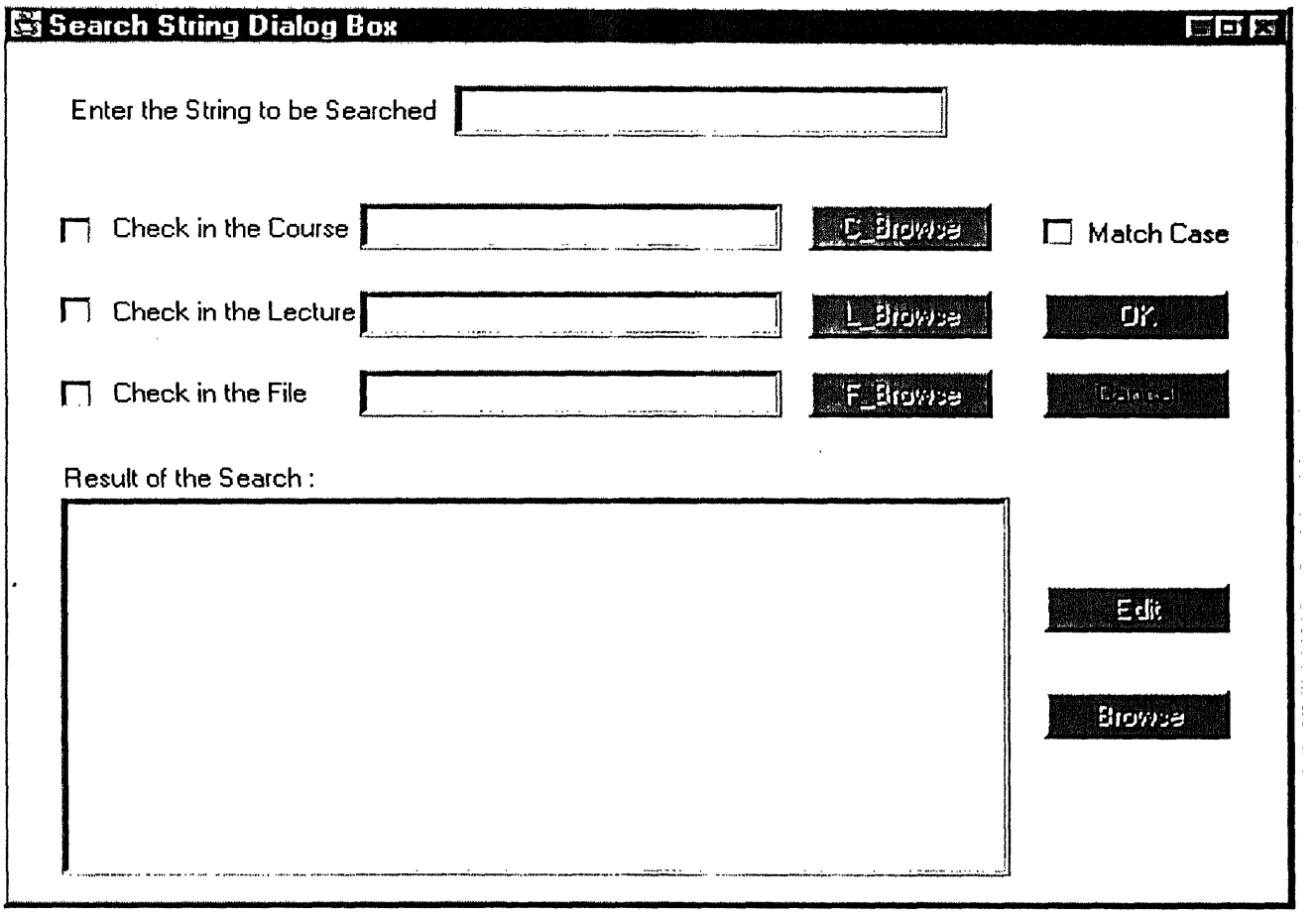

Figure 8: Searching a Keyword

It can happen that the specified string may be present in more than one file so the names of the files should be shown to the user. In order to do that a linked list is formed where each node contains the name of the file, the line number and column number where the string is present. This linked list is shown to the user as the entries to a list, from here onwards referred as graphical list. Direct calls to the presentation system like Netscape and the HTML editor is also provided to allow the user to actually either view the hyperboards or edit them. The structure of each node is given below :-

```
found_object {
      String objname;
      int line_position;
      int col_position;
}
```

Here *objname* contains the name of the searched hyperboard with full path, line position and column position.

Other submodules in this module are :-

- Searchfile
- Open
- Openlecture
- Opencourse
- Check
- InsertElement
- Display

These submodules are explained in appendix A.

4.10 Organization of a New Lecture

This module is used to organize the contents of a lecture. A lecture as already explained contains a list of hyperboards. In this module the user is given the option of adding the hyperboard file into a lecture file. Along with this CUT,COPY and PASTE options are also provided for the user to move around the hyperboards within the lecture file. Selection of a particular hyperboard is done through a file dialog box, with this box the user can navigate through the various system directories and select the exact hyperboard file which he wants to place in the lecture file. The user interface is shown in figure 9.

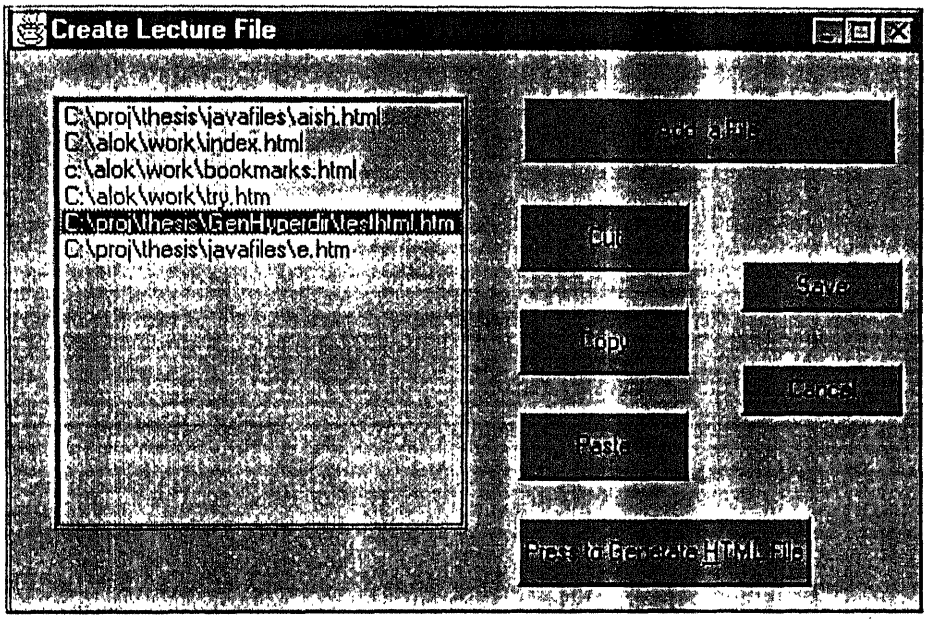

Figure 9: Creation of a Lecture

The hyperboard currently present in the lecture file are shown as file names along with their full path in the graphical list. This module maintains a linked list, whose individual nodes contain the names of the individual files. The linked list helps us to allow movement of individual nodes easily within it. Structure of each node is given below :-

```
lecture_object {
     String objname;
     }
```

Here the *objname* contains the name of the file. Other submodules in this module are :-

- Add a File
- InsertElement
- DeleteElement
- Display

- CutElement
- PasteElement
- SaveFile
- SaveHtmlFile

These submodules are explained in appendix A.

4.11 Modify Lecture

This module is used to modify the contents of an exiting lecture file. It provides a file dialog box for the user to pick-up the desired lecture file from the system's directory and then calls the overloaded constructor of the create lecture module explained in the previous section. This constructor creates an empty linked list for the keywords, opens the disk file containing the lecture and calls fill module to read the file and fill the linked list. Rest of the program is same as explained in the previous section.

4.12 Organization of a New Course

This module is used to organize the contents of a course. A course as already explained contains a list of lecture file. In this module the user is given the option of adding the lecture file into a course file. Along with this CUT,COPY and PASTE options are also provided for the user to move around the lecture files within the course file. Selection of a particular lecture file is done through a file dialog box. The user interface is shown in figure 10.

The lecture files currently present in the course file are shown as the name of the file along with its full path in the graphical list. This module maintains a linked list whose individual nodes contains the name of the individual files. The linked list easily allows movement of individual nodes within it. The structure of each node is given below :-

```
course {
```

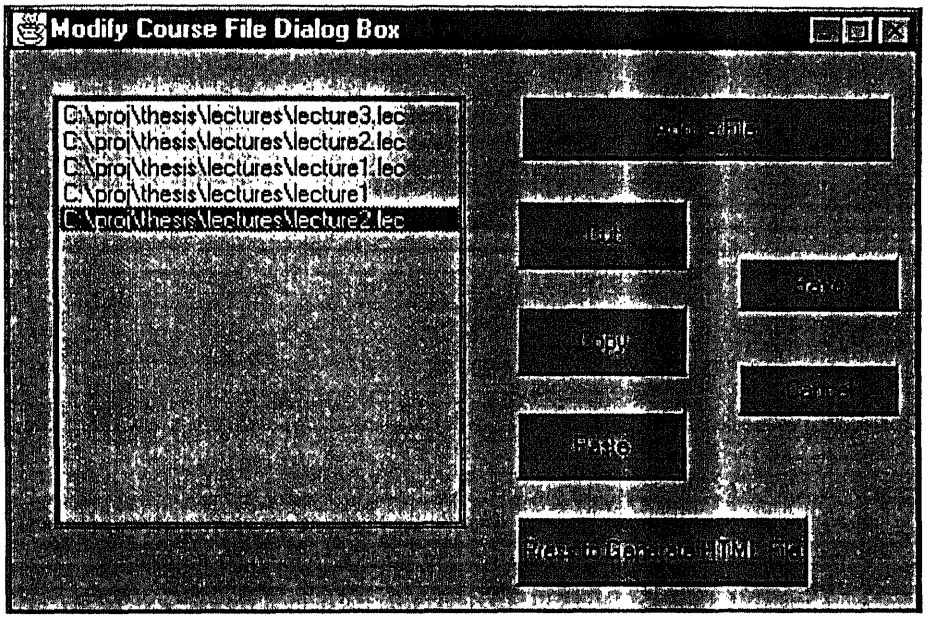

Figure 10: Creation of a Course

```
String objname;
}
```

Here the *objname* contains the name of the file. Other submodules in this module are :-

- Add a File
- InsertElement
- DeleteElement
- Display
- CutElement
- PasteElement
- SaveFile
- SaveHtmlFile

These submodules are explained in appendix A.

4.13 Modify Course

This module is used to modify the contents of the exiting course file. It provides a file dialog box for the user to pick-up the desired course file from the system's directory and then calls the overloaded constructor of the create course module explained in the previous section. This constructor creates an empty linked list for the keywords, opens the disk file containing the course and calls fill module to read the file and fill the linked list. Rest of the program is same as explained in the previous section.

4.14 View File

This module is used to view the contents of a course as a whole. It shows a list of all the lecture files along with the list of all the hyperboards in each lecture. This gives the overall view of the whole course to the user. This module also provides the facility to edit the course file if required, which involves either removing a hyperboard from the lecture file or moving it from one lecture to another.

For viewing the individual hyperboards the user is provided with direct calls to a presentation system like Netscape and also to the HTML editor. This module opens a window which is divided into two frames one showing the contents of the course in a tree like structure and the other frame is used to show the hyperboard the user wishes to see. Each lecture file is shown with a "+" sign in the tree indicating that it contains a list of hyperboards. However, this list can be seen by the user by clicking the mouse button at the lecture filename, on doing so the name explodes into a list of hyperboards present in the lecture file and the sign changes to "-". The user is provided with a file dialog box to select the course file to view. The user interface is shown in figure 11.

This module creates a linked list in which each node either represents the name of the lecture file or the hyperboard. The structure of each node is given below :-

```
lecture_view_object {
    String objname;
    String path;
    String state;
```

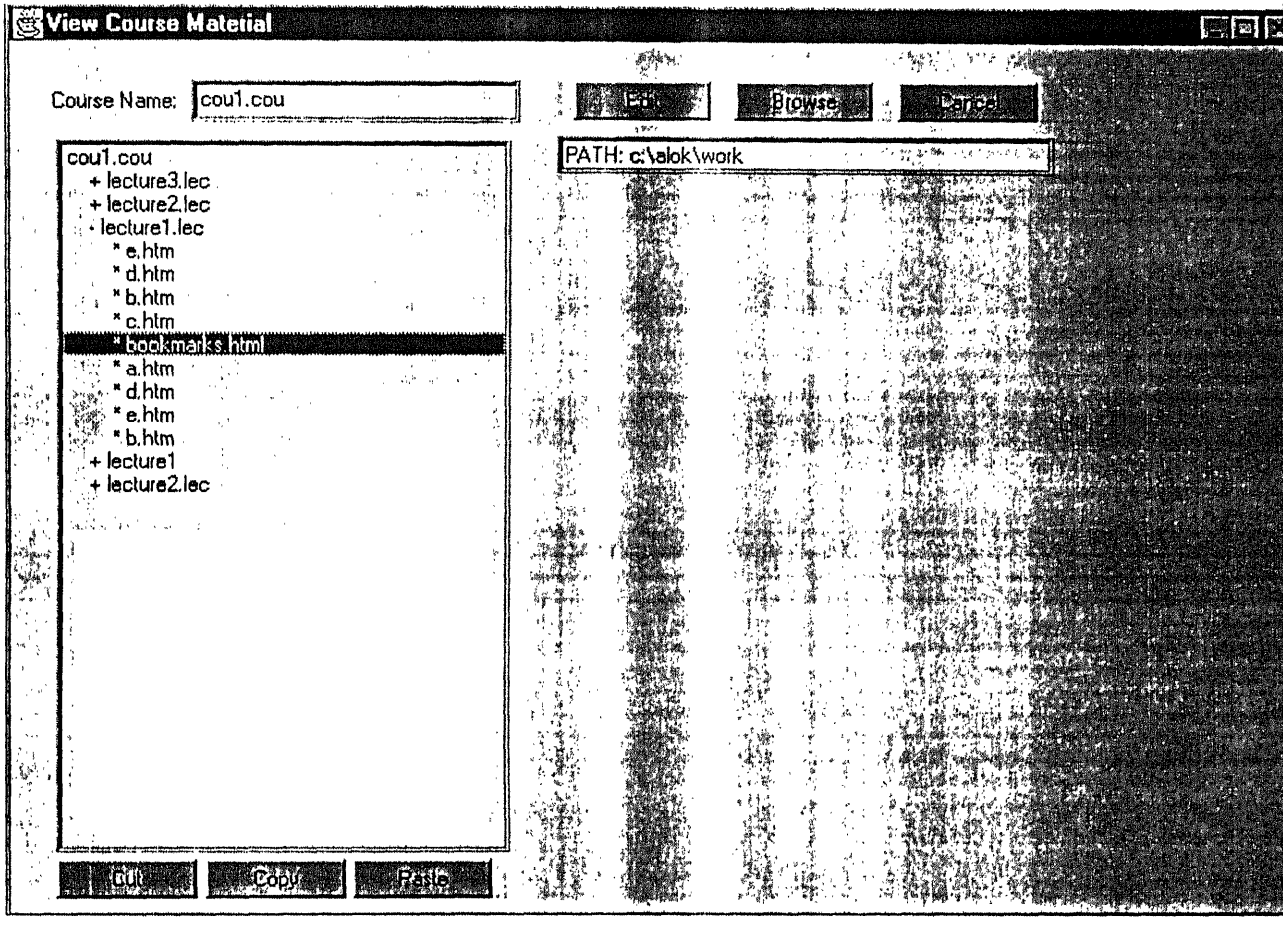

Figure 11: Viewing a Course

}

Here *objname* is the name of the lecture file or the hyperboard, *path* represents the full path of the file and the *state* has the following values :-

- "+" if the lecture file contains the list of hyperboards but not shown.
- "-" if the lecture file is showing the list of hyperboards .
- "*" used for the hyperboard file meaning that it cannot be further decomposed.

Other submodules used in this module are :-

- Open

- Fill
- AddSpace
- InsertElement
- DeleteElement
- Display
- OpenForInsert
- FillForInsert
- CreateFile
- CutElement
- PasteElement
- SaveFile

These submodules are explained in appendix A.

4.15 Help

This module provides on-line help on our authoring system. For the creation of on-line help Microsoft Visual Help software was used. This software requires us to create the necessary text for the help in the Rich Text Format (RTF) in which the various topics are defined along with their explanations. This RTF text was prepared using Microsoft Word Ver 7.0. This text was then compiled by the MS VHelp software to produce a help file in .HLP format. This help file can be viewed using WinHelp.exe of MS Windows-95. Thus in our system whenever the user wishes to use the help, this module calls the *RunCommand* submodule which creates a separate process and executes WinHelp.exe with our Help file in .HLP format as the argument. The user interface is shown in figure 12.

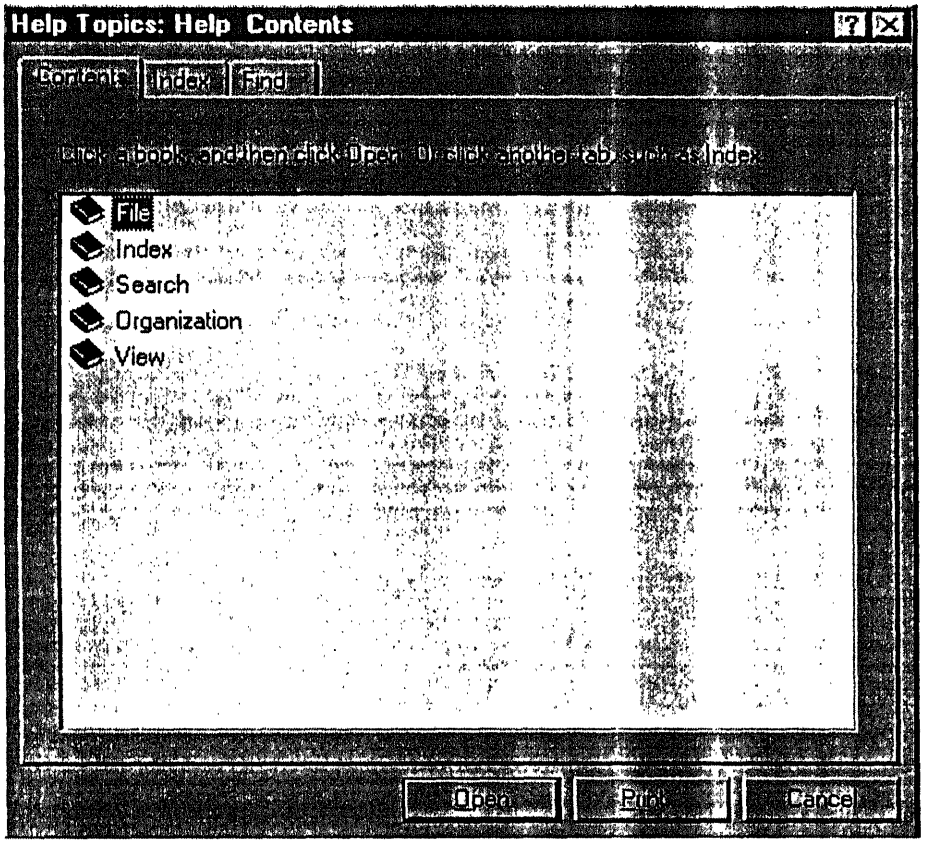

Figure 12: User Help

Chapter 5

Conclusion

The authoring tool with all its features, as decided in the design phase, was implemented using the java programming language provided through the Microsoft Developer Studio. A case study has been done in a comparison thesis in which a course on programming in Java was developed using this tool. Netscape Ver 4.0 was used as the presentation system and a stand-alone HTML editor for Windows was also used. The whole course was developed as a set of lectures and each lecture was designed as a set of hyperboards. All the features present in our system were rigorously tested during the development of the course.

5.1 Future Work

The authoring system developed is a core authoring system for creating hyperboard based courses. A full featured authoring system will need the following additional functions :-

- **Plug-Ins:** The capability of the tool can be greatly enhanced if certain type of plug-in tools are added. Plug-ins can be provided for designing tests, quizzes, question banks, animation etc. related to a particular course. These plug-ins can be both general and in certain cases specific to the type of subject. To cite an example, Suppose the Instructor is taking a course in mathematics then it would be a great help to him if he is given a tool by which he can show the

effect of integration on a function through animation.

- **Typing of Links:** In the current system all the links used during the creation of the hyperboard are navigation links that takes the user from one place to another in the same or a different document. Some more links can be added in the system such as animation links, links that can draw diagrams etc. These links can be incorporated in the hyperboard during its creation. Thus when the hyperboard is shown in the class the instructor will know that on clicking a specific link he can draw a diagram or can animate the concept etc.

- **Version Control:** As of now our system does not offer the facility for the user to maintain different versions of course material. The author has to maintain the different versions manually by placing the course material in different system directories. Version control needs to be automated. A facility to embed multiple versions in the same basic information space has to be added. If a new Instructor wishes to teach the same subject he can see the old version and can improve upon it without disturbing it. Also, tools have to be provided by which the two versions can be compared and only the changes stored in later versions.

- **Scheduling of Lectures:** Tools are needed by which the Instructor is able to schedule the lecture in terms of their duration once they are created. This scheduling should depend on the amount of time the instructor has. Whenever he designs a new hyperboard he should specify its relative importance such as (*important, very important, not very important*) , so that if he has less time for a course during the preparation of the lecture only the important and very important hyperboards are added.

- **Level of Instruction:** The instructor should be given the facility to decide the level of each hyperboard such as *beginner, intermediate and advanced level.* This will help in designing lectures for different audiences.

Bibliography

[AGR95] D B Lowe A Ginige and J Robertson. Hypermedia authoring. *IEEE MultiMedia Winter*, pages 24–35, Winter 1995.

[App] Authoring Tool Approaches. URL. *http://www.c3.lanl.gov/village/ui/subsubsection2_2_*

[Gos95] James Gosling. *The Java Programming Language Vol-I,II,III.* Addison Willay, 1995.

[Hol96] Steven Holzner. *Java WorkShop.* BPB, 1996.

[iaAt] What is an Authoring tool. URL. *http://www.c3.lanl.gov/village/ui/subsubsection2_2_1_1.htm.*

[J.87] Conklin J. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, Sept 1987.

[lvCL] Scripting/Visual language vs Code Libraries. URL. *http://www.c3.lanl.gov/village/ui/subsubsection2_2_2_1_1.htm.*

[NS96] Patrick Naughton and Herbert Schildt. *Java Complete Reference.* Osborne Mc Graw Hill, 1996.

[NYD88] N K Meyrowitz N Yankelovich, B J Haan and S M Drucker. Intermedia: The concept and the construction of a seamless information system. *IEEE Computer*, pages 81–96, January 1988.

[NYV85] N. Meyrowitz N Yenkelovich and A Vandam. Reading and writing the electronic book. *IEEE Computer*, 18(10):15-30, Oct 1985.

[Proa] Easy Computer Based Teaching Program. URL. *http://www.easyteach.com/easycbt.htm.*

[Prob] Everest Computer Based Teaching Program. URL. *http://www.insystem.com/evunique.htm.*

[Sys] Evaluation System. URL. *http://www.c3.lanl.gov/village/ui/subsubsection2_2_2_2.htm.*

[Too] Eon Knowledge-Based Tutor Authoring Tools. URL. *http://www.cs.umass.edu/ ckc/eon/eon.html.*

Appendix A

Submodules Description

This chapter gives the detailed description of the various submodules that are present in the tools described in the implementation chapter.

A.1 Creating a New HyperBoard

The various submodules used in this module are :-

- InsertIntoVector
- GetInfo
- Convert
- SaveFile

A.1.1 InsertIntoVector

This submodule opens the selected stylesheet disk file and the reads it line by line to separate each component along with its properties. For each line a parser is run which reads the line and produces tokens that are the properties of the component. Once all the properties of the component are found this module calls the InsertElement submodule to actually form a node and put it in the linked list.

A.1.2 GetInfo

This submodule reads the text keyed in the components placed in the input form window. These components are textfields and textareas. Textfield can contain a single line of text while a textarea can hold multiple lines of text.

A.1.3 Convert

This is a very small submodule whose basic job is to convert all the linefeed characters keyed in the textarea to the linefeed character of HTML.

A.1.4 SaveFile

This submodule generates the HTML file for the text keyed in the input form. This submodule traverses the linked list to get the properties of each component and then generate suitable HTML tags for them before placing the actual text which the user has keyed-in. This submodule before actually saving the file provides a file dialog box for the user to select the directory in which he wishes to place the file.

A.2 Creating an Index

The various submodules in this module are :-

- Fill
- InsertMainElement
- DisplayMainlist
- InsertSubcomp
- DisplaySubcomp
- CutElement
- PasteElement

- SaveFile
- SaveHtmlFile

A.2.1 Fill

This submodule is executed as soon as the constructor of the main index module is executed. This submodule creates the main linked list of the alphabetic characters. There all the keywords are arranged in the alphabetical order. Thus this module only creates the main linked list with no linked list attached to any of these alphabetic characters.

A.2.2 InsertMainElement

This submodule forms a node of the new keyword and adds it to the main linked list. The keyword is placed at the right position depending upon the place where the user has decided to place it. For placing the keyword, first the user has to mark the place in the existing list and can then key-in the keyword.

A.2.3 DisplayMainlist

This submodule continuously monitors the main linked list and shows it in the left frame of the index creation window.

A.2.4 InsertSubcomp

This submodule is used to add the references to the keyword added through the InsertMainElement submodule. This submodule shows a file dialog box to the user to select the correct hyperboard that contains the keyword. After the selection is made this submodule forms a node containing the path information of the hyperboard and adds it to the linked list attached to the node representing this keyword in the main linked list.

A.2.5 DisplaySubcomp

This submodule shows the list of references for a particular keyword in the right hand frame of the index creation window. As soon as the user clicks on the particular keyword to see its references this submodule reads the linked list attached to this keyword and displays it on the right hand frame.

A.2.6 CutElement

This submodule is used to either remove or move a keyword from one place to another in the index. If the keyword has to be removed then this submodule removes its corresponding node from the main linked list along with its references linked list. To remove the keyword from the main linked list it call the deleteElement submodule. If the keyword has to be moved from one place to another then this submodule creates a temporary node coping all the details of the node in it along with its references linked list. It then inserts this temporary node at the new place in the main linked list and removes the old node.

A.2.7 PasteElement

This submodule is used to copy a keyword from one place to several other places in the main linked list. This submodule first creates a temporary node of the keyword that has to be copied along with its references linked list. It then copies this temporary node to all the other places where they have to be copied.

A.2.8 SaveFile

This submodule creates a disk file of the index created. It traverses each node of the main linked list and also the references linked list attached to each node and copies the keyword and its references to the disk file.

A.2.9 SaveHtmlFile

This submodule creates a HTML file for the index. Thus even the index file forms a hyperboard. Here each keyword is represented as simple text with all its references represented as links to the hyperboards that contain it. These references are serially numbered in the way they were added during index creation.

A.3 Open Index

The various submodules in this modules are :-

- InsertMainElement
- InsertSubcomp
- DisplayMainlist
- DisplaySubcomp

All these modules work in the same way as they work in the create index module.

A.4 Search

The various submodules in this modules are :-

- Searchfile
- Open
- Openlecture
- Opencourse
- Check
- InsertElement
- Display

A.4.1 SearchFile

This submodule, depending upon the checkbox selection by the user, decides whether the search is to be carried out on a single hyperboard or lecture file or the course file and calls the relevant submodule with the name of the hyperboard file or the lecture file or the course file as the argument.

A.4.2 Open

This submodule opens the file to be searched. It calls the check submodule which will carry out the search.

A.4.3 Openlecture

This submodule opens the lecture file. Since a lecture file contains a list of hyperboard files this submodule reads each hyperboard file and calls the Open submodule to carry our the search.

A.4.4 Opencourse

This submodule opens the course file. Since a course file contains a list of lecture files this submodule reads each lecture file and calls the Openlecture submodule to carry our the search which in turn calls the Open submodule.

A.4.5 Check

This submodule reads the individual lines of the hyperboard sent to it as the argument and puts them in a buffer. Now it calls the Java *indexof* function of the class String to find out the index of the string, to be searched, in the buffer. If the search is successful the return is the column number where the string is present in the buffer and if unsuccessful the return is -1. Depending upon the result it calls the InsertElement submodule to form a node and insert it in the linked list.

A.4.6 InsertElement

This submodule gets the name of the filename, the line number and the column number from the check submodule, forms a node and inserts it in the linked list.

A.4.7 Display

This submodule reads the linked list and presents the contents of each node as a separate entry to the graphical list present in the lower half of the window.

A.5 Organization of a New Lecture

The various submodules in this modules are :-

- Add a File
- InsertElement
- DeleteElement
- Display
- CutElement
- PasteElement
- SaveFile
- SaveHtmlFile

A.5.1 Add a File

This submodule opens a file dialog box for the user to navigate and pickup the required hyperboard file for the lecture file. This submodule then calls the InsertElement submodule that adds this name to the linked list.

A.5.2 InsertElement

This submodule on getting the name of the file as the argument forms a node and adds it to the linked list.

A.5.3 Display

This submodule reads the linked list and presents the contents of each node as a separate entry to the graphical list present in the left half of the window.

A.5.4 DeleteElement

This submodule removes an entry from the linked list.

A.5.5 CutElement

This submodule is used to either remove or move a hyperboard from one place to another in the lecture file. If the hyperboard has to be removed then this submodule removes its corresponding node from the main linked list. To remove the hyperboard from the main linked list it call the deleteElement submodule. If the hyperboard has to be moved from one place to another then this submodule creates a temporary node coping all the details of the node in it. It then inserts this temporary node at the new place in the main linked list and removes the old node.

A.5.6 PasteElement

This submodule is used to copy a hyperboard from one place to several other places in the main linked list. This submodule first creates a temporary node of the hyperboard that has to be copied and then copies it to all the other places.

A.5.7 SaveFile

This submodule creates a disk file of the lecture created. It traverses each node of the main linked list and copies each node to the disk file.

A.5.8 SaveHtmlFile

This submodule creates a HTML file for the lecture. Thus even the lecture file forms a hyperboard. Here each hyperboard is represented as a number with its full path represented as link to the hyperboard . These references are serially numbered in the same way they were added during the lecture creation.

A.6 Organization of a New Course

The various submodules in this module are :-

- Add a File
- InsertElement
- DeleteElement
- Display
- CutElement
- PasteElement
- SaveFile
- SaveHtmlFile

A.6.1 Add a File

This submodule opens a file dialog box for the user to navigate and pickup the required lecture file for the course file. This submodule then calls the InsertElement submodule that adds this name to the linked list.

A.6.2 InsertElement

This submodule on getting the name of the file as the argument forms a node and adds it to the linked list.

A.6.3 Display

This submodule reads the linked list and presents the contents of each node as a separate entry to the graphical list present in the left half of the window.

A.6.4 DeleteElement

This submodule removes an entry from the linked list.

A.6.5 CutElement

This submodule is used to either remove or move a lecture file from one place to another in the course file. If the lecture file has to be removed then this submodule removes its corresponding node from the main linked list. To remove the lecture file from the main linked list it calls the *deleteElement* submodule. If the lecture file has to be moved from one place to another then this submodule creates a temporary node coping all the details of the node in it. It then inserts this temporary node at the new place in the main linked list and removes the old node.

A.6.6 PasteElement

This submodule is used to copy a lecture file from one place to several other places in the main linked list. This submodule first creates a temporary node of the lecture file that has to be copied and then copies it to all the other places.

A.6.7 SaveFile

This submodule creates a disk file of the course created. It traverses each node of the main linked list and copies the lectures to the disk file.

A.6.8 SaveHtmlFile

This submodule creates a HTML file for the course. Thus even the course file forms a hyperboard. Here each lecture is represented as a number with its full path

represented as link to the lecture file. These references are serially numbered in the same way they were added during the course creation.

A.7 View File

The various submodules used in this module are :-

- Open
- Fill
- AddSpace
- InsertElement
- DeleteElement
- Display
- OpenForInsert
- FillForInsert
- CreateFile
- CutElement
- PasteElement
- SaveFile

A.7.1 Open

This submodule opens the course disk file to be read by the Fill submodule.

A.7.2 Fill

This submodule reads the course file opened by the Open submodule. It reads this file line by line and calls the AddSpace submodule.

A.7.3 AddSpace

This submodule creates a linked list, gets the file name as the argument from the Fill submodule, forms the node and adds it to the linked list. The initial state of all the nodes is "+". This submodule is also called when the lecture file is expanded into its hyperboards wherein it creates the node of individual filenames with a "*" sign.

A.7.4 InsertElement

This submodule is used whenever the user clicks the mouse button on any lecture filename. In that case the lecture file is read and its contents, that is, the hyperboards are displayed in the graphical list. This module calls OpenForInsert and FillForInsert submodules with the name of the lecture file as the argument.

A.7.5 OpenForInsert

This submodule opens the lecture disk file to be read by the FillForInsert submodule.

A.7.6 FillForInsert

This submodule reads the lecture file opened by the OpenForInsert submodule. It reads this file line by line and calls the AddSpace submodule which adds the hyperboard filename to the linked list.

A.7.7 Display

This submodule continuously reads the linked list and displays a tree like structure in the graphical list present in the left frame of the window.

A.7.8 DeleteElement

This submodule is used when the mouse button is clicked by the user at the expanded lecture file with "-" state. In that case the expanded tree of the lecture file is collapsed and all the hyperboard files which are shown in the graphical list is removed. This

is done by physically removing the nodes from the linked list and calling the Display submodule.

A.7.9 CreateFile

Whenever the user wants to view the contents of any hyperboard a presentation system like Netscape should show the file within the right hand frame of the window. Netscape window size is not within the control of the user, so a small Java-script was written that can move the Netscape window within that frame after invocation. This submodule generates that java-script in a temporary file and calls Netscape with this temporary file as the argument.

A.7.10 CutElement

This submodule is used to either remove a hyperboard from the lecture file or move it to another lecture file. In this case when the hyperboard file is to be removed from the lecture file, first the node of the hyperboard is removed from the linked list and the SaveFile submodule is called which reads the linked list and saves a fresh copy of the lecture file so that the change is permanent.

In case the hyperboard file is to be moved from one place to another then a temporary node of this hyperboard is created and PasteElement submodule is called which then places it to the new position.

A.7.11 PasteElement

This submodule is called after the Cut operation, its job is to place the temporary node created by the CutElement submodule to a new place in the linked list. To make the change permanent it also opens the respective lecture file where this hyperboard is to be placed and calls SaveFile to save it.

A.7.12 SaveFile

This submodule reads the portion of the linked list related to any lecture file and saves its contents.